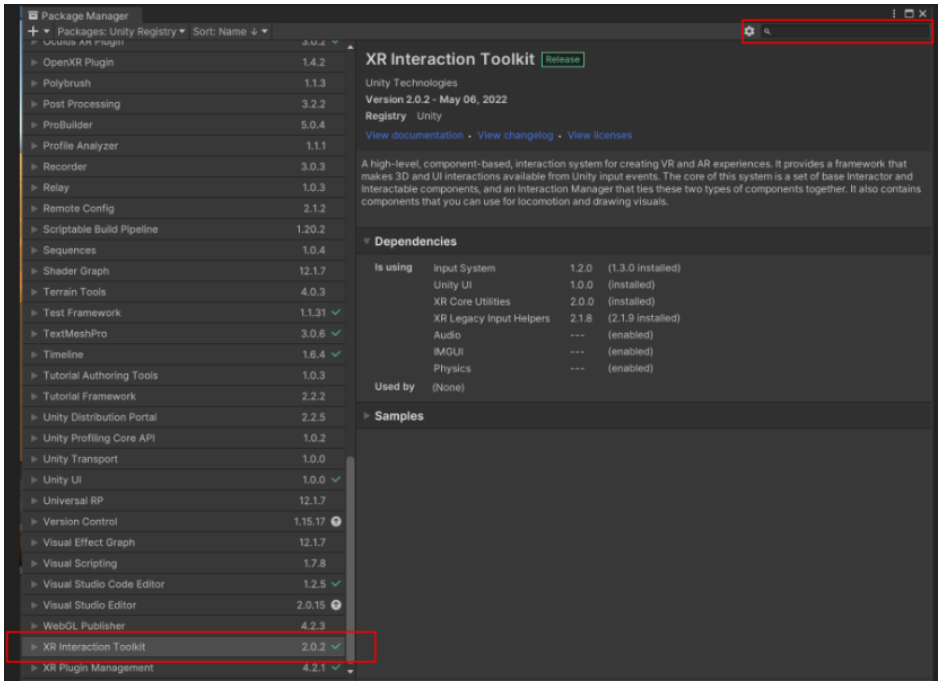
Experiment No. 6
Develop a simple UI(User interface) menu with images, canvas, sprites and button. Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction display a score on scene .
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Develop a simple UI(User interface) menu with images, canvas, sprites and button. Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction display a score on scene .

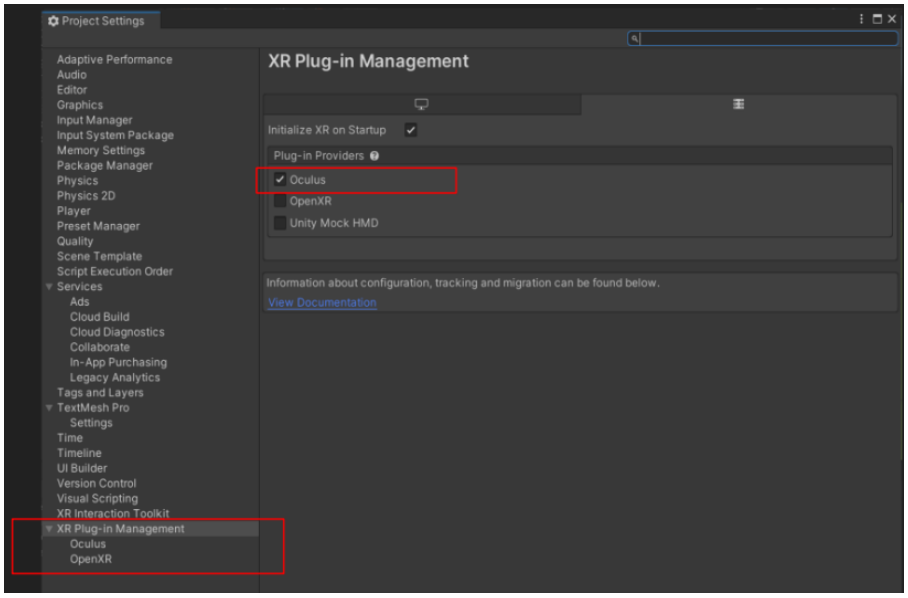
Theory:

VR by using XR Interaction Toolkit:
1. Unity has created a package called "XR Interaction Toolkit" by which we can create VR enabled applications for a wide range of VR Headsets.
2. To install the XR Interaction Toolkit package go to Window -> Package Manager.
3. Click on the "Packages:Unity Project" dropdown in the top left corner & select "Unity Registry".
4. Click on the plus icon on the top left corner of the package manager window & click "Add package by name" & type the following:
com.unity.xr.interaction.toolkit
5. After installation, update the package if necessary.
6. Additionally search & install these packages if they aren't already installed:
○ XR Plugin Management
○ Oculus XR Plugin
7. Unity might ask to restart the project, click yes if it does.



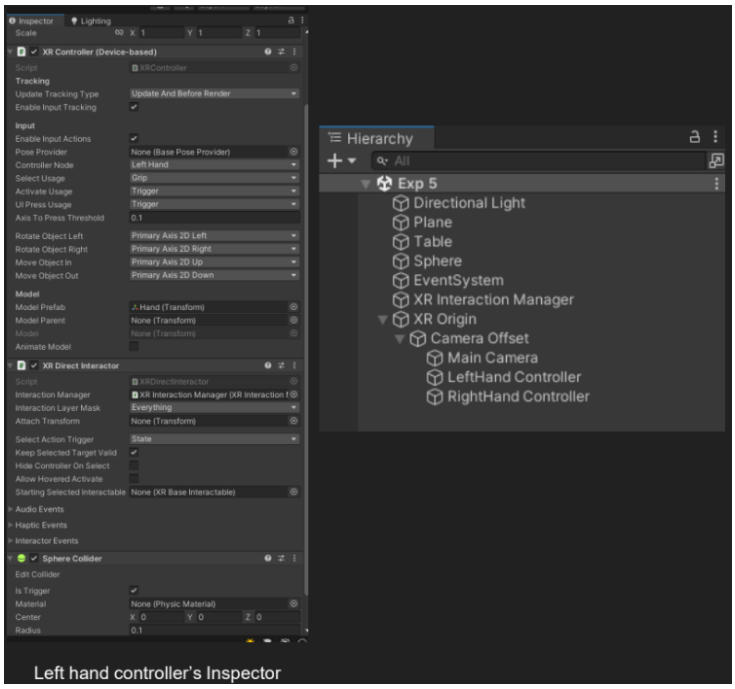
VR by using XR Interaction Toolkit:
1. Now close the package manager window & go to Edit -> Project settings.
2. Click on XR Plug In Management Tab in the bottom left.

3. On the right side, Check the oculus checkbox as said in the Image.



Creating the XROrigin:

1. The XR Origin represents the center of worldspace in an XR scene.
2. The purpose of the XR Origin is to transform objects and trackable features to their final position, orientation, and scale in the Unity scene. It specifies an Origin, a Camera Floor Offset Object, and a Camera.
3. In the hierarchy, right click -> XR -> Device Based -> XROrigin.
4. Set the Transform position of the XROrigin at 0, 0, 0. Set the Tracking Origin Mode field of the XR Origin component to Floor.
5. Expand the hierarchy of XROrigin (Tip: left arrow besides the object in Hierarchy).
6. Select the LeftHand Controller & remove the XR Ray Interactor, XR Interactor Line Visual, Line Renderer components from it. Add XR Direct Interactor & Sphere Collider to it. Set the Sphere Collider values acc to the image. Do the same for the RightHand Controller.



Left hand controller's Inspector

Writing the GameManager Script:

1. Create a new Folder "Experiment 6" & then a folder called "Scenes" inside it. Duplicate Exp 5 scene by clicking on it in the Project window & pressing Ctrl + D. Remove the sphere & table gameobjects.
2. Move the Exp 6 scene to the Experiment 6's Scenes folder.
3. Create a "GameManager" script inside the Experiment 6 -> Scripts folder.
4. Double click to open it in Visual Studio.
5. Write the script according to the image:
6. Read the comments to understand the script.

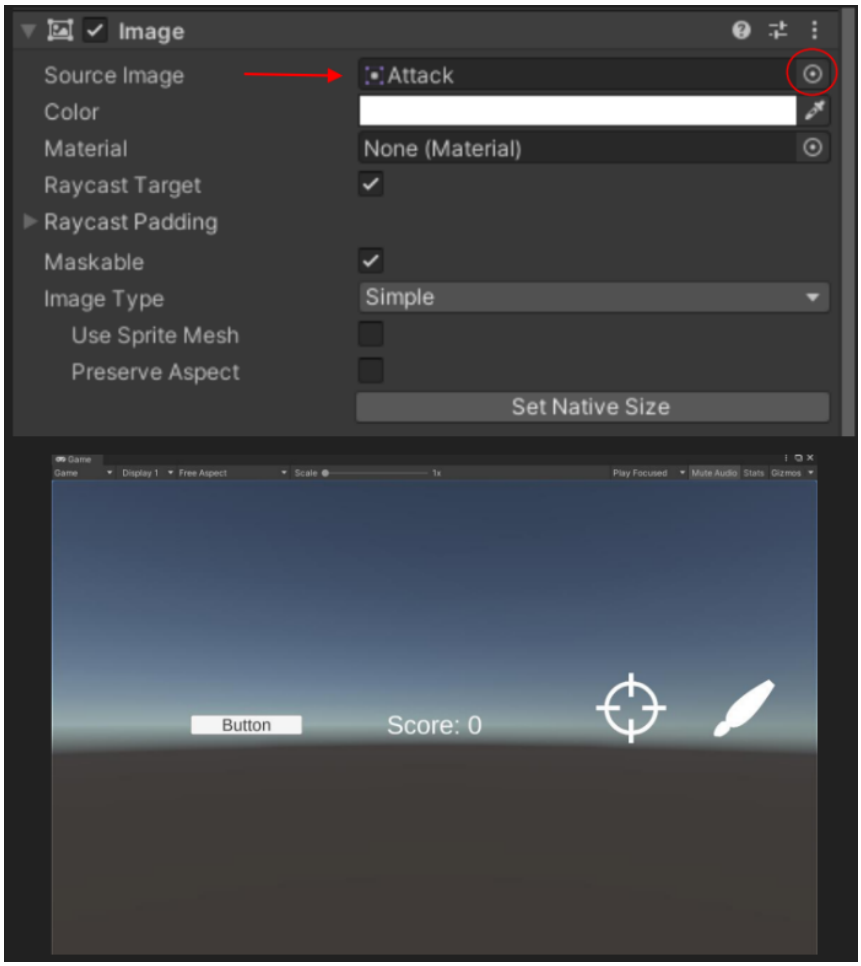
```
GameManager.cs
Assembly-CSharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

namespace Exp6
{
    public class GameManager : MonoBehaviour
    {
        [Tooltip("The score text component where we are displaying the Score")]
        public TextMeshProUGUI scoreText;

        [Tooltip("Current Score")]
        public int currentScore = 0;

        [Tooltip("The amount with which the score should increase on each Click")]
        public int scoreIncrement = 10;

        /// <summary>
        /// Called on the first frame.
        /// </summary>
        private void Start()
        {
            SetScoreText();
        }

        /// <summary>
        /// Called on each UI Click
        /// </summary>
        public void OnUIClick()
        {
            currentScore += scoreIncrement;
            SetScoreText();
        }

        /// <summary>
        /// Set the score according to a format.
        /// </summary>
        void SetScoreText()
        {
            scoreText.text = "Score: " + currentScore;
        }
    }
}
```

Creating the UI:

1. In the hierarchy, right click in the empty area UI -> Canvas.
2. Create a panel game object as a child of it after right clicking on the Canvas gameobject. Set the color of the panel to black with the alpha (transparency) set to 0.5.
3. Now create a Button - TextMeshPro & Text - TextMeshPro under the Canvas gameObject.
4. Create 2 image gameObjects. (UI -> Images) under the Canvas gameObject.
5. The hierarchy should look like this:

Positioning the UI Elements:

1. Set the text of the TextMeshPro - Text to "Score : 0".
2. Create a folder named "Images" under the experiment's folder in the Project view. To set the images of the 2 image gameobjects, import 2 images from outside the Unity project into the project.
3. Click on all the images in the Images folder & set it's texture type to "Sprite (2D and UI)".
4. Now in the hierarchy, select one of the Image gameObjects & set it's Source Image property to the new images imported by clicking on the small circle besides the field & then searching for the image. Do the same for the other image gameobject.
5. Set the position of the buttons & Images through it's RectTransform to look like the Game view image below.



Interacting with the UI Elements:

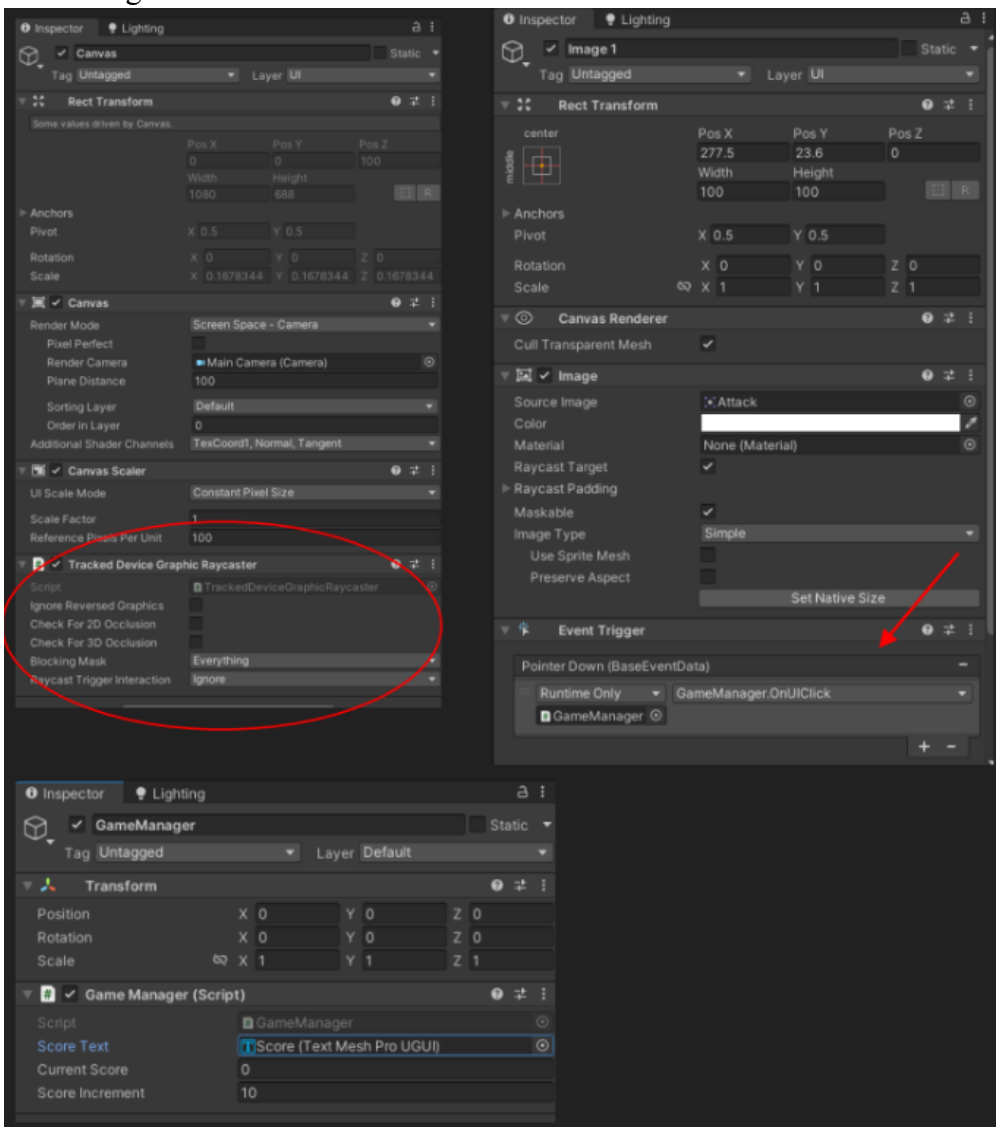
1. To interact with the UI elements, click on the canvas gameobject & in the inspector window. Then right click on the Graphic Raycaster component & select Remove Component. Add Tracked Graphic Raycaster to it.
2. Create a new empty GameObject in the hierarchy by right click-> Create Empty. Name it "GameManager".
3. Add the script we created earlier - GameManager as a component to it. Set ScoreText field to

the Score gameobject.

4. Select the 2 Image GameObjects & Add Component- Event Trigger. Select a single image gameobject. Click the "Add New Event Type" button on the bottom of the Event Trigger component & select Pointer Down option. Click on the plus icon on the bottom of Pointer Down event.

5. Drag & drop the GameManager from the hierarchy into the "None (Object)" field created. From the dropdown select GameManager -> OnUIClick()

6. Do the same thing for the button OnClick event.

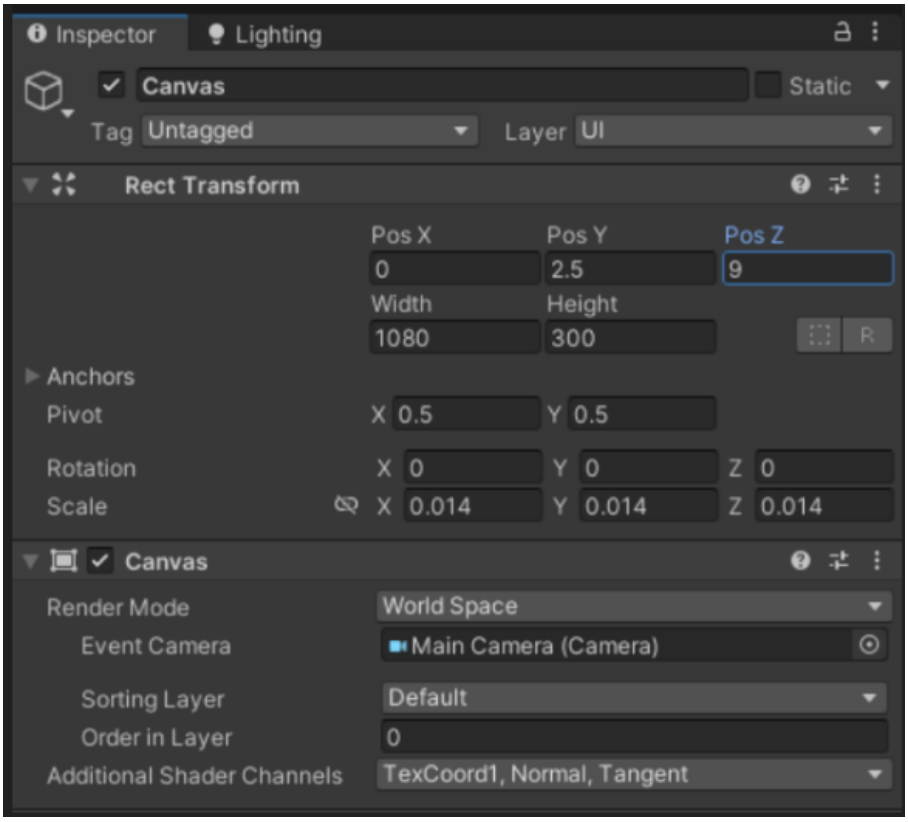


World Space Canvas:

1. Currently the Canvas is in screen space, meaning it is used as an overlay on whatever we see from the camera.

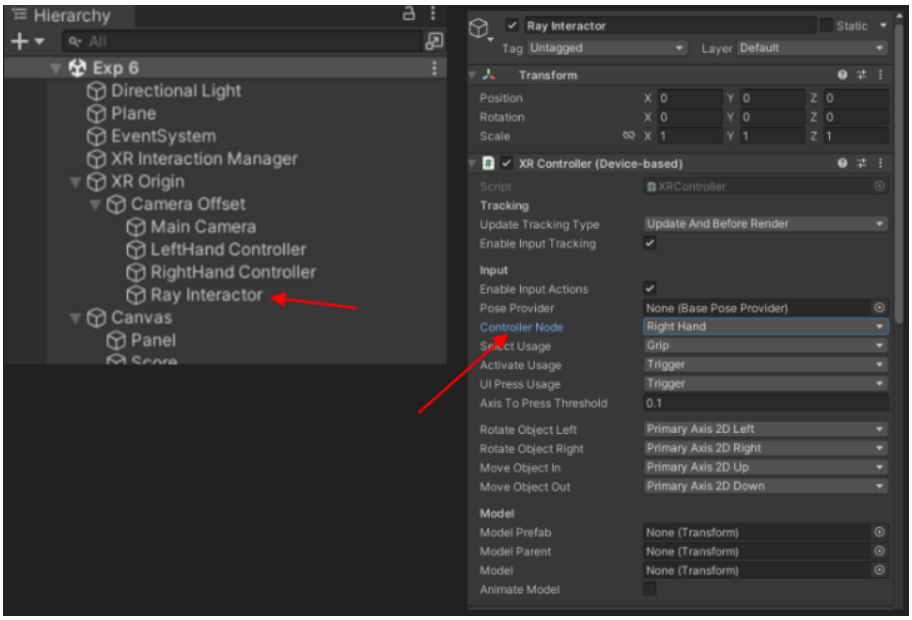
2. But this cannot work in VR mode. So to fix that we have to change the Render mode of the Canvas component to World Space.

3. Set the RectTransform values to the following so the canvas is visible in front when we see it in Play mode in VR.



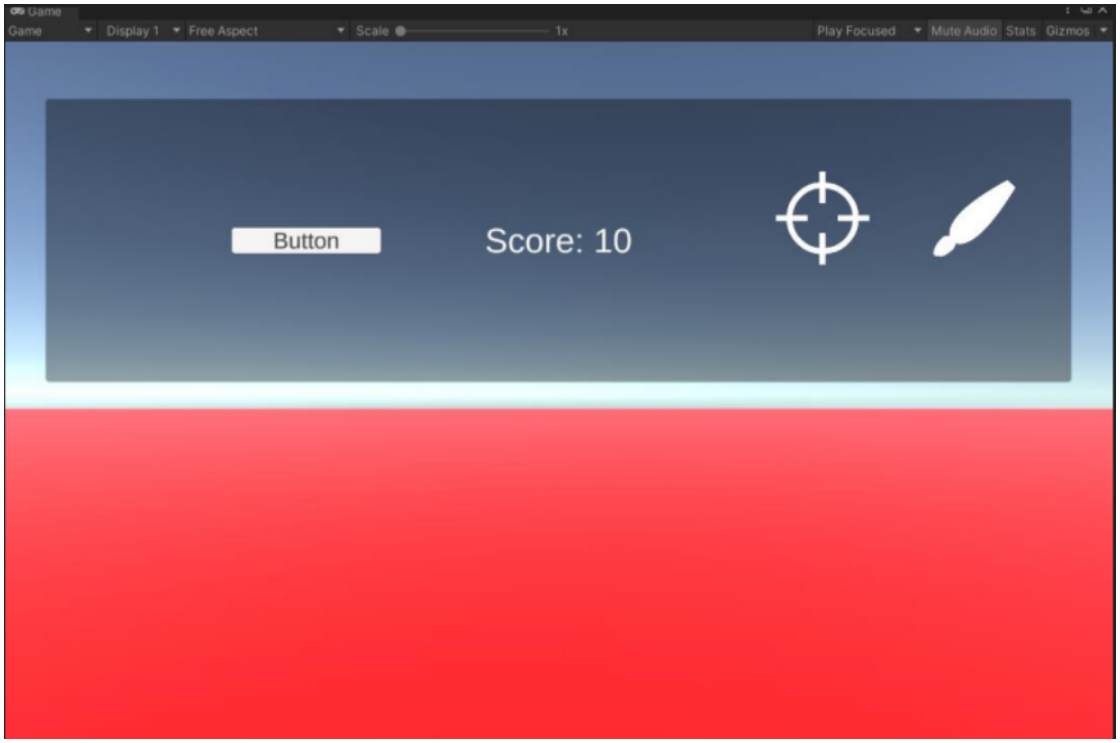
Ray Interactors:

1. In the hierarchy, click on the small arrow on the left side XROrigin gameobject to expand it.
2. Right click on the Camera Offset -> XR -> Device based -> Ray Interactor.
3. Scroll Down in the inspector & change the Line Width field value of XR Interactor Line Visual to 0.02.
4. Select the EventSystem gameobject from the hierarchy & remove the Standalone UI Module & add XR UI Input module.



Play the Game:
1. Play the Game & Click on the UI Elements - buttons & images with the help of the trigger button on the right controller.
2. Watch the score increase on the Canvas.

Output:



Conclusion:-

In Unity, we successfully created a user-friendly UI menu with images, canvas, sprites, and buttons, providing an immersive and interactive interface. We developed a C# program that enabled VR trigger button interactions, and on each successful interaction, a score was displayed within the scene. This experiment effectively showcases the integration of virtual reality with user interfaces, enhancing user engagement and providing valuable feedback through dynamic scoring. Such applications have the potential to enhance VR experiences, gamification, and interactive simulations, opening up a range of possibilities for immersive content creation and user engagement.